Lab 8

1. An *AVL Tree* is a BST that satisfies a different balance condition, namely:

   The AVL Balance Condition For each internal node x, the height of the left child of x differs from the height of the right child of x by at most 1. (Equivalently, the heights of the left and right subtrees of x differ by at most 1.)

   Create a red-black tree that does *not* satisfy the AVL Balance Condition.

2. Use the insertion algorithm for red-black trees to successively insert the following nodes, starting with an empty tree.

   a. 1, 2, 3, 4, 5, 6, 7, 8
   b. 3, 2, 1, 5, 4, 6

   Note on Part (a): Recall that an already sorted insertion sequence is a worst case for an ordinary BST. Notice how the red-black balancing operations handle this to remain balanced.

3. Devise an algorithm IsPrime(*n*) which outputs TRUE if n is prime, FALSE otherwise. Then implement as a Java method. What is the asymptotic running time of IsPrime? Explain.

4. In the course, we have determined asymptotic running times of sorting algorithms as a function of input size *n.* However, in number-theoretic algorithms, such as GCD, the running time has been bounded by functions of *n* where, in this case, *n* is an input *value*, but does not represent the input *size.* The reason is that the *size* of a natural number *n*, from the point of view of any reasonable computational model, is *its length as a bit string* and not simply the value *n* itself.

   Examples:
   If n = 7, its size as the bit string 111 is 3.
   If n = 67, its size as the bit string 1000011 is 7.

   In general, $length(n) = \lfloor \log n \rfloor + 1$.

   When running times of number-theoretic algorithms are expressed in terms of input *size* rather than input *value* (as we have done so far), results can appear unfamiliar.

   For example, the asymptotic running time of GCD(*m*,*n*) in terms of input values, as we have seen, is O(log *n*). However, since *n* is $O(2^{length(n)})$, in terms of input *size*, GCD(*m*,*n*) runs in O(*length*(*n*)). That is, GCD(*m*,*n*) is *linear in the size of n.* Here is a more careful analysis:

GCD Algorithm

```
Algorithm GCD(m,n)
Input nonnegative integers m, n, not both 0
Output gcd(m,n)
if n=0 then
  return m
else
  return GCD(n, m % n)
```

(**) In general, if T(n) is O(f(n)), in terms of the *value n*, then, in terms of the size b = b(n) of input n, T(b) is O(f(2^b)). In the case of GCD, since gcd(m,n) runs in O(log n), in terms of the value n, gcd runs in O(log 2^b) = O(b) in terms of input size.

In light of the above discussion, answer the following:

A. Express the asymptotic running time of your algorithm IsPrime(*n*) in terms of the input *size* rather than input value. It may be helpful to use two arguments, *n*, *b*(*n*), to help focus on the number of bits of n when computing running time; then you can compute running time *T*(*b*) in terms of the input size. Or you can simply compute the running time in terms of n, then convert to running time in terms of b using the formula given in (**) above.
B. Suppose *T*(*b*) is the running time of your algorithm in terms of input size. Show that b^2 is o(*T*(*b*)).